

**S.Y.B.Sc**

**Computer Science**

**2019 – 2020**

**This is to certify that PRASHANT SINGH of S.Y.B.Sc Roll No. 70 has successfully completed the practical of Paper – III (sem – III)** **Operating System during the Academic Year 2019-2020 as specified by the MUMBAI UNIVERSITY.**

**Prof. (Mrs.) Vaidehi Deshpande Prof. (Mrs.) Anita Gaikwad**

**Professor-In-charge In-charge**

**Dept. of Computer Science Dept. of Computer Science**

**Name : PRASHANT SINGH**

**Roll\_No: 70**

**Subject: Paper III (Operating System)**

**Class: S.Y.B.Sc (COMPUTER SCIENCE)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **INDEX** | | | | |
| **NO** | **DATE** | **TITLE** | **PAGE NO** | **SIGN** |
|  | | | | |
| 1 |  | Give the solution to producer-consumer problem. |  |  |
| 2 |  | Write a java program for synchronization. |  |  |
| 3 |  | Implement FCFS scheduling algorithm in java |  |  |
| 4 |  | Implement SJF (No Preemption) scheduling algorithm in java. |  |  |
| 5 |  | Implement RR scheduling algorithm in java. |  |  |
| 6 |  | Write a java program that implements the banker’s algorithm.  Safety  Recourse Request |  |  |
| 7 |  | Write a java program to implement disk scheduling algorithm (SSTF). |  |  |
| 8 |  | Write a java program to implement Optimal page replacement algorithm of memory management. |  |  |

**PRACTICAL-1**

**Aim:** Give the solution to producer -consumer problem .

**Source Code-**

class Inventory

{

static int qoh=500;

static int req=0;

static public synchronized void request(int order)

{

if(order<=qoh)

{

System.out.println("quantity ordered:"+order);

qoh-=order;

req+=order;

System.out.println("quantity on hand:"+qoh);

System.out.println("Total quantity taken away by way of order:"+req);

}

else

{

System.out.println("ordered quantity more than quantity on hand");

}

}

public static void main(String args[])

{

new TestThread();

try

{

for(int p=3;p>0;p--)

{

System.out.println("-----------");

System.out.println("main thread:"+p);

System.out.println("-----------");

Thread.sleep(1000);

}

}

catch(InterruptedException e)

{

}

System.out.println("Exiting main thread...");

}

}

class TestThread extends Thread

{

TestThread()

{

super("Test Thread");

System.out.println("child Thread:"+this);

start();

}

public void run()

{

for(int i=5;i>0;i--)

{

try

{

sleep(100);

}

catch(InterruptedException e)

{

System.out.println("i="+i);

}

Inventory.request((int)(Math.random()\*100));

}

}

}

**Output:**

child Thread:Thread[Test Thread,5,main]

-----------

main thread:3

-----------

quantity ordered:7

quantity on hand:493

Total quantity taken away by way of order:7

quantity ordered:48

quantity on hand:445

Total quantity taken away by way of order:55

quantity ordered:36

quantity on hand:409

Total quantity taken away by way of order:91

quantity ordered:76

quantity on hand:333

Total quantity taken away by way of order:167

quantity ordered:61

quantity on hand:272

Total quantity taken away by way of order:228

-----------

main thread:2

-----------

-----------

main thread:1

-----------

Exiting main thread...

**PRACTICAL-2**

**Aim:** Write a JAVA program for Synchronization.

**Source Code-**

**Stock.java**

class Consumer implements Runnable

{

Stock c;

Thread t;

Consumer(Stock c)

{

this.c=c;

t=new Thread(this,"consumer Thread");

t.start();

}

public void run()

{

while(true)

{

try

{

t.sleep(750);

}

catch(Exception e)

{

}

c.getStock((int)(Math.random()\*1000));

}

}

void stop()

{

t.stop();

}

}

class Producer implements Runnable

{

Stock s;

Thread t;

Producer(Stock s)

{

this.s=s;

t=new Thread(this,"Producer Thread");

t.start();

}

public void run()

{

while(true)

{

try

{

t.sleep(750);

}

catch(Exception e)

{

}

s.addStock((int)(Math.random()\*1000));

}

}

void stop()

{

t.stop();

}

}

public class Stock

{

int goods=0;

public synchronized void addStock(int i)

{

goods=goods+i;

System.out.println("stock added:"+i);

System.out.println("present stock:"+goods);

notify();

}

public synchronized int getStock(int j)

{

while(true)

{

if(goods>=j)

{

goods=goods-j;

System.out.println("Stock taken away:"+j);

System.out.println("present stock:"+goods);

break;

}

else

{

System.out.println("Stock not enough");

System.out.println("Waiting for stock to come");

try

{

wait();

}

catch(Exception e)

{

}

}

}

return goods;

}

public static void main(String args[])

{

Stock j=new Stock();

Producer p= new Producer(j);

Consumer c=new Consumer(j);

try

{

Thread.sleep(10000);

p.stop();

c.stop();

System.out.println("Thread stopped");

}

catch(Exception e)

{

}

System.exit(0);

}

}

**Output:**

stock added:75

present stock:75

Stock taken away:17

present stock:58

stock added:537

present stock:595

Stock taken away:20

present stock:575

stock added:325

present stock:900

Stock taken away:719

present stock:181

stock added:145

present stock:326

Stock not enough

Waiting for stock to com

stock added:340

present stock:666

Stock taken away:575

present stock:91

stock added:30

present stock:121

Stock not enough

Waiting for stock to com

stock added:441

present stock:562

Stock taken away:218

present stock:344

stock added:631

present stock:975

Stock taken away:639

present stock:336

stock added:783

present stock:1119

Stock taken away:345

present stock:774

stock added:336

present stock:1110

Stock taken away:221

present stock:889

stock added:345

present stock:1234

Stock taken away:222

present stock:1012

stock added:898

present stock:1910

Stock taken away:413

present stock:1497

stock added:423

present stock:1920

Stock taken away:857

present stock:1063

Thread stopped

**PRACTICAL-3**

**Aim:** Implement FCFS scheduling algorithm in java.

**Source Code-**

**Fcfs.java**

import java.io.\*;

class Fcfs

{

public static void main(String args[]) throws Exception

{

int n,AT[],BT[],WT[],TAT[];

float AWT=0;

InputStreamReader isr=new InputStreamReader(System.in);

BufferedReader br=new BufferedReader(isr);

System.out.println("Enter no of process");

n=Integer.parseInt( br.readLine());

BT=new int[n];

WT=new int[n];

TAT=new int[n];

AT=new int[n];

System.out.println("Enter Burst time for each process\n");

for(int i=0;i<n;i++)

{

System.out.println("Enter BT for process"+(i+1));

BT[i]=Integer.parseInt( br.readLine());

}

//System.out.println("\*\*\*\*\*\*\*\*");

for(int i=0;i<n;i++)

{

System.out.println("Enter AT for process"+i);

AT[i]=Integer.parseInt( br.readLine());

}

//System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

WT[0]=0;

for(int i=1;i<n;i++)

{

WT[i]=WT[i-1]+BT[i-1];

WT[i]=WT[i]-AT[i];

}

for(int i=1;i<n;i++)

{

TAT[i]=WT[i]+BT[i];

AWT=AWT+WT[i];

}

System.out.println("PROCESS BT WT TAT");

for(int i=1;i<n;i++)

{

System.out.println(" "+i+" "+BT[i]+" "+TAT[i]);

}

AWT=AWT/n;

//System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Avg waiting time="+AWT);

}

}

**Output:**

Enter no of process

2

Enter Burst time for each process

Enter BT for process1

3

Enter BT for process2

3

Enter AT for process0

2

Enter AT for process1

2

PROCESS BT WT TAT

1 3 4

Avg waiting time=0.5

**PRACTICAL-4**

**Aim:** Implement SJF scheduling algorithm in java.

**Source Code-**

**TestClass.java**

import java.util.Scanner;

class TestClass

{

public static void main(String args[])

{

int burst\_time[],process[],waiting\_time[],tat[],i,j,n,total=0,pos,temp;

float wait\_avg,TAT\_avg;

Scanner s=new Scanner(System.in);

System.out.println("Enter number of process");

n=s.nextInt();

process=new int[n];

burst\_time=new int[n];

waiting\_time=new int[n];

tat=new int[n];

System.out.println("\n Enter Burst Time:");

for(i=0;i<n;i++)

{

System.out.println("\nProcess["+(i+1)+"]:");

burst\_time[i]=s.nextInt();

process[i]=i+1;

}

for(i=0;i<n;i++)

{

pos=i;

for(j=i+1;j<n;j++)

{

if(burst\_time[j]<burst\_time[pos])

pos=j;

}

temp=burst\_time[i];

burst\_time[i]=burst\_time[pos];

burst\_time[pos]=temp;

temp=process[i];

process[i]=process[pos];

process[pos]=temp;

}

waiting\_time[0]=0;

for(i=1;i<n;i++)

{

waiting\_time[i]=0;

for(j=0;j<i;j++)

waiting\_time[i]+=burst\_time[j];

total+=waiting\_time[i];

}

wait\_avg=(float)total/n;

total=0;

System.out.println("\nProcess\t Burst Time\t waiting Time\t Turnaround Time");

for(i=0;i<n;i++)

{

tat[i]=burst\_time[i]+waiting\_time[i];

total+=tat[i];

System.out.println("\n p"+process[i]+"\t\t"+burst\_time[i]+"\t\t"+waiting\_time[i]+"\t\t"+tat[i]);

TAT\_avg=(float)total/n;

System.out.println("\n\nAverage Waiting Time:"+wait\_avg);

System.out.println("\nAverage Turnaround Time:"+TAT\_avg);

}

}

}

**Output:**

Enter number of process

5

Enter Burst Time:

Process[1]:

1

Process[2]:

2

Process[3]:

2

Process[4]:

3

Process[5]:

3

Process Burst Time waiting Time Turnaround Time

p1 1 0 1

Average Waiting Time:3.4

Average Turnaround Time:0.2

p2 2 1 3

Average Waiting Time:3.4

Average Turnaround Time:0.8

p3 2 3 5

Average Waiting Time:3.4

Average Turnaround Time:1.8

p4 3 5 8

Average Waiting Time:3.4

Average Turnaround Time:3.4

p5 3 8 11

Average Waiting Time:3.4

Average Turnaround Time:5.6

**PRACTICAL-5**

**Aim:** Implement RR (Round Robin) scheduling algorithm in java.

**Source Code-**

**OS1.java**

import java.io.\*;

class Job implements Runnable

{

int process\_id, no\_of\_instr,time\_quantum;

Thread t;

job(int pid, int instr, int tq)

{

process\_id = pid;

no\_of\_instr = instr;

time\_quantum = tq;

t = new Thread(this);

t.start();

}

public void run()

{

try

{

for(int i=1; i<=no\_of\_instr; i++)

{

System.out.println("Executing instr no " + i + " of process " + process\_id);

Thread.sleep(time\_quantum);

}

System.out.println("Job " + process\_id + " is over");

}

catch(InterruptedException e)

{

System.out.println("The job has been interrupted...");

}

}

}

class OS1

{

public static void main(String args[])

{

try

{

int process\_id=100, time\_quantum = 100;

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter a user process starting number:");

process\_id = Integer.parseInt(br.readLine());

System.out.println("Enter a time quantum(in millis):");

time\_quantum = Integer.parseInt(br.readLine());

job j1 = new job(++process\_id, 10, time\_quantum);

job j2 = new job(++process\_id, 6, time\_quantum);

job j3 = new job(++process\_id, 8, time\_quantum);

}

catch(Exception e)

{

System.out.println("Some process failed to complete...");

System.out.println("Plz contact system admin...");

}

}

}

**Output:**

Enter a user process starting number:

500

Enter a time quantum(in millis):

100

Executing instr no 1 of process 501

Executing instr no 1 of process 502

Executing instr no 1 of process 503

Executing instr no 2 of process 502

Executing instr no 2 of process 501

Executing instr no 2 of process 503

Executing instr no 3 of process 502

Executing instr no 3 of process 501

Executing instr no 3 of process 503

Executing instr no 4 of process 502

Executing instr no 4 of process 501

Executing instr no 4 of process 503

Executing instr no 5 of process 502

Executing instr no 5 of process 501

Executing instr no 5 of process 503

Executing instr no 6 of process 502

Executing instr no 6 of process 501

Executing instr no 6 of process 503

Job 502 is over

Executing instr no 7 of process 501

Executing instr no 7 of process 503

Executing instr no 8 of process 501

Executing instr no 8 of process 503

Executing instr no 9 of process 501

Job 503 is over

Executing instr no 10 of process 501

Job 501 is over

**PRACTICAL-6**

**Aim:** Write a java program that implements the banker’s algorithm. Safety Recourse Request.

**Source Code-**

**OS2.java**

import java.io.\*;

class Banker

{

int avail[];

int max[][] = { {3,2,2,1}, {8,12,0,0}, {2,1,0,0}, {4,3,0,0}, {2,0,3,1}};

int alloc[][] = { {1,1,1,0}, {2,1,0,0}, {1,0,0,0}, {2,1,0,0}, {1,0,0,0}};

int need[][];

int m,n;

Banker()

{

m = 4;

n = 5;

avail = new int[4];

avail[0] = 16; // Number of Registers

avail[1] = 50; // Number of Files

avail[2] = 5; // Number of Ports

avail[3] = 2; // Number of Printer

need = new int[5][4];

for(int i=0;i<5; i++)

for(int j=0;j<4; j++)

need[i][j] = max[i][j] ‐ alloc[i][j];

}

boolean isSafe()

{

int work[] = new int[m];

boolean finish[] = new boolean[n];

for(int i=0 ;i<m; i++)

work[i] = avail[i];

for(int i=0 ;i<n; i++)

finish[i] = false;

for(int i=0; i<n; i++)

{

if(finish[i]==false)

{

boolean incomplete=false;

for(int j=0;j<m; j++)

{

if(need[i][j]>work[j])

{

incomplete = true;

//System.out.println(i + ":" + j + ":" + need[i][j] + ":" + work[j]);

}

if(need[i][j]!=0 && need[i][j]<=work[j])

{

work[j] = work[j] ‐ need[i][j];

}

}

if(!incomplete)

{

finish[i]=true;

}

if(finish[i])

System.out.println("Process " + i + " can be completed");

else

System.out.println("Process " + i + " can't be completed");

}

}

for(int i=0; i<n; i++)

{

if(!finish[i])

{

return false;

}

}

return true;

}

void resourceRequest()

{

int request[][] = { {1,1,1,0}, {2,2,0,0}, {1,1,0,1}, {1,1,0,0}, {1,0,0,0}};

boolean safe=true;

for(int i=0; i<n; i++)

{

for(int j=0;j<m; j++)

{

if(request[i][j]<=need[i][j])

{

if(request[i][j]<=avail[j])

{

avail[j] = avail[j] ‐ request[i][j];

alloc[i][j] = alloc[i][j] + request[i][j];

need[i][j] = need[i][j] ‐ request[i][j];

}

else

{

System.out.println("Hey process " + i + "! resource " + j + " is not available now...try again

later...");

}

}

else

{

System.out.println("Process " + i + " has exceeded request for resource " + j + " hence is

unsafe");

safe = false;

}

}

}

if(safe)

{

System.out.println("the system is safe...");

}

else

{

System.out.println("the system is unsafe...");

}

}

}

class OS2

{

public static void main(String args[]) throws Exception

{

Banker b = new Banker();

System.out.println("Applying safety algorithm...");

if(b.isSafe())

{

System.out.println("The system is in safe state...");

}

else

{

System.out.println("The system is not in safe state....");

}

System.out.println("Applying resource request algo...");

b.resourceRequest();

}

}

**Output-**

Applying safety algorithm...

Process 0 can be completed

Process 1 can be completed

Process 2 can be completed

Process 3 can be completed

Process 4 can be completed

The system is in safe state...

Applying resource request algo...

Process 2 has exceeded request for resource 3 hence is unsafe

the system is unsafe...

**PRACTICAL-7**

**Aim:** write a java program to implement Disk scheduling algorithm(SSTF).

**Source Code-**

**OS3.java**

import java.io.\*;

class DevMgmt

{

String disk\_sequence;

DevMgmt(String fs)

{

disk\_sequence = fs;

}

void sstf(int start\_track)

{

System.out.println("======= SHORTEST SEEK‐TIME FIRST ===========");

String disknums[] = disk\_sequence.split(" ");

int tracks[] = new int[disknums.length];

boolean tracks\_bool[] = new boolean[disknums.length];

int min\_dist=0,max\_dist=0;

for(int i=0; i<disknums.length; i++)

{

tracks[i] = Integer.parseInt(disknums[i]);

tracks\_bool[i] = false;

max\_dist = max\_dist+tracks[i];

}

min\_dist = max\_dist;

int total\_cylinders=0,from\_track=start\_track,current\_track\_index=‐1;

System.out.println("Traversing from:" + from\_track);

for(int i=0; i<tracks.length; i++)

{

int to\_track\_index=0;

for(int t=0; t<tracks.length;t++)

{

if((!tracks\_bool[t]) && (Math.abs(from\_track‐tracks[t])<min\_dist) && (t!=current\_track\_index) )

{

to\_track\_index=t;

min\_dist = Math.abs(from\_track‐tracks[t]);

}

}

System.out.println("to track : " + tracks[to\_track\_index] );// ":" + min\_dist);

total\_cylinders+= Math.abs(from\_track‐tracks[to\_track\_index]);

tracks\_bool[to\_track\_index] = true;

from\_track = tracks[to\_track\_index];

current\_track\_index=to\_track\_index;

min\_dist=max\_dist;

}

System.out.println("Total cylinders traversed = " + total\_cylinders);

}

}

class OS3

{

public static void main(String args[]) throws Exception

{

String disk\_sequence = "98 183 37 122 14 124 65 67";

int track = 53;

DevMgmt m = new DevMgmt(disk\_sequence);

m.sstf(track);

}}

**Output-**

Enter disk frame sequence(separated by space):

Enter scan scheduling algo:

Enter starting track number:

======= SHORTEST SEEK‐TIME FIRST ===========

Traversing from:53

to track : 65

to track : 67

to track : 37

to track : 14

to track : 98

to track : 122

to track : 124

to track : 183

Total cylinders traversed = 236

**PRACTICAL-8**

**Aim:** Write a java program to implement Optimal page replacement algorithm of **memory management .**

**Source Code-**

**OS4.java**

import java.io.\*;

class MemMgmt

{

String frame\_sequence;

int mem\_block[];

MemMgmt(String fs,int n)

{

frame\_sequence = fs;

mem\_block = new int[n];

for(int i=0;i<n; i++)

mem\_block[i]=‐1;

}

void dispMemBlock()

{

System.out.print("|");

for(int i=0;i<mem\_block.length;i++)

{

System.out.print(mem\_block[i]+"|");

}

System.out.println();

}

void fcfs()

{

System.out.println("======= FCFS ===========");

String strpages[] = frame\_sequence.split(" ");

int pages[] = new int[strpages.length];

for(int i=0; i<strpages.length; i++)

pages[i] = Integer.parseInt(strpages[i]);

int mem\_block\_num=0, page\_faults=0;

System.out.println("Initial Memory layout...");

dispMemBlock();

for(int i=0; i<pages.length; i++)

{

boolean present=false;

for(int j=0;j<mem\_block.length;j++)

{

if(mem\_block[j] == pages[i])

{

present=true;

break;

}

}

if(!present)

{

mem\_block[mem\_block\_num] = pages[i];

mem\_block\_num++;

page\_faults++;

}

if(mem\_block\_num==mem\_block.length)

mem\_block\_num=0;

System.out.println("Loading page no." + (i+1) + ":" + pages[i]);

dispMemBlock();

}

System.out.println("Total Number of page faults:" + page\_faults);

}

void oppr()

{

System.out.println("======= Optimal Page Replacement ===========");

String strpages[] = frame\_sequence.split(" ");

int pages[] = new int[strpages.length];

int i=0;

for(i=0; i<strpages.length; i++)

pages[i] = Integer.parseInt(strpages[i]);

int mem\_block\_num=0,page\_faults=0;

System.out.println("Initial Memory layout...");

dispMemBlock();

for(i=0; i<mem\_block.length; i++)

{

boolean present=false;

for(int j=0;j<mem\_block.length;j++)

{

if(mem\_block[j] == pages[i])

{

present=true;

break;

}

}

if(!present)

{

mem\_block[mem\_block\_num] = pages[i];

mem\_block\_num++;

page\_faults++;

}

System.out.println("Loading page no." + (i+1) + ":" + pages[i]);

dispMemBlock();

}

for(; i<pages.length; i++)

{

boolean present=false;

for(int j=0;j<mem\_block.length;j++)

{

if(mem\_block[j] == pages[i])

{

present=true;

break;

}

}

if(!present)

{

mem\_block\_num=‐1;

int longest\_page=‐1;

for(int j=0;j<mem\_block.length; j++)

{

int k=0;

for(k=i+1; k<pages.length; k++)

{

if(mem\_block[j] == pages[k])

{

if(k>longest\_page)

{

longest\_page = k;

mem\_block\_num = j;

}

break;

}

}

if(k==pages.length)

{

longest\_page = pages.length;

mem\_block\_num = j;

}

}

mem\_block[mem\_block\_num] = pages[i];

page\_faults++;

}

System.out.println("Loading page no." + (i+1) + ":" + pages[i]);

dispMemBlock();

}

System.out.println("Total Number of page faults:" + page\_faults);

}

}

class OS4

{

public static void main(String args[]) throws Exception

{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter page frame sequence(separated by space):");

//String frame\_sequence = br.readLine();

String frame\_sequence = "7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1";

System.out.println("Enter page replacement algo:");

int i = Integer.parseInt(br.readLine());

MemMgmt m = new MemMgmt(frame\_sequence,3);

if(i==1)

m.fcfs();

else

if(i==2)

m.oppr();

}

}

**Output-**

Enter page frame sequence(separated by space):

Enter page replacement algo:

======= Optimal Page Replacement ===========

Initial Memory layout...

|‐1|‐1|‐1|

Loading page no.1:7

|7|‐1|‐1|

Loading page no.2:0

|7|0|‐1|

Loading page no.3:1

|7|0|1|

Loading page no.4:2

|2|0|1|

Loading page no.5:0

|2|0|1|

Loading page no.6:3

|2|0|3|

Loading page no.7:0

|2|0|3|

Loading page no.8:4

|2|4|3|

Loading page no.9:2

|2|4|3|

Loading page no.10:3

|2|4|3|

Loading page no.11:0

|2|0|3|

Loading page no.12:3

|2|0|3|

Loading page no.13:2

|2|0|3|

Loading page no.14:1

|2|0|1|

Loading page no.15:2

|2|0|1|

Loading page no.16:0

|2|0|1|

Loading page no.17:1

|2|0|1|

Loading page no.18:7

|7|0|1|

Loading page no.19:0

|7|0|1|

Loading page no.20:1

|7|0|1|

Total Number of page faults:9